

ROUTING MISBEHAVIOR AND PERFORMANCE ANALYSIS OF DSR ALGORITHM IN MANET

*Dissertation submitted to Jawaharlal Nehru University, in partial fulfillments of the
requirements for award of the degree of*

Master of Technology

in

Computer Science and Technology

by

GONELLA V.S. RAMA SANKAR



**SCHOOL OF COMPUTER & SYSTEMS SCIENCES
JAWAHARLAL NEHRU UNIVERSITY
NEW DELHI – 110067**

January 2002

005.1

TH

R1405 Ro



TH9441





जवाहरलाल नेहरू विश्वविद्यालय
JAWAHARLAL NEHRU UNIVERSITY
School of Computer & Systems Sciences
NEW DELHI - 110 067, INDIA

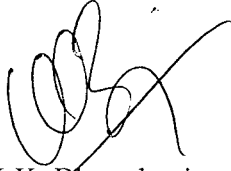
CERTIFICATE

This is to certify that the project entitled “ **Routing Misbehavior and Performance Analysis of DSR algorithm in MANET** ” being submitted by **Gonella V. S. Rama Sankar** to the School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi, in partial fulfillment of the requirements for the award of the degree of Master of Technology in Computer Science & Technology, is a bonafide work carried out by him under the guidance and supervision of Prof. G.V.Singh and Dr. D.K.Lobiyal.

The matter embodied in the dissertation has not been submitted for the award of any other degree or diploma.


Prof. G.V.Singh
Professor, SC & SS,
Jawaharlal Nehru University
New Delhi – 67


Dr. D.K. Lobiyal
Asst. Professor, SC & SS
Jawaharlal Nehru University
New Delhi - 67


Prof. K.K. Bharadwaj
Dean, SC & SS,
Jawaharlal Nehru University,
New Delhi-110067.

**...dedicated to
my beloved parents**

ACKNOWLEDGEMENTS

I would like to pay obeisance at the feet of my parents for their blessings that are always with me in all my aspirations including my academics.

I would like to sincerely thank my supervisors Prof. G.V. Singh, Dr D.K. Lobiyal, School of Computer And System Sciences, Jawaharlal Nehru University for the help, encouragement and support extended by them in successful completion of this project. Their ideas were innovative and the valuable discussions we had were very much helpful in keeping the project on the right track.

I would like to record my sincere thanks to the dean, Prof. K.K. Bharadwaj for providing the necessary lab facilities.

I extend my sincere gratitude to my lab mates for their continuous academic as well as morale support through out my dissertation work.

I will acknowledge the efforts and knowledge put by the development team of VINT project team at UC Berkeley, for developing the network simulator and making its source code available on the Internet as a freeware.

Last, but not the least, I take this opportunity to thank all the faculty members and friends for their, help and encouragement during the course of the project.

(Gonella V.S. Rama Sankar)

ABSTRACT

The present work studies the performance of the dynamic source routing algorithm in the context of misbehaving nodes. The network throughput depends on the how efficiently the underlying protocol transfer the data across the nodes of the network. If the nodes of the network forward the packets properly then algorithm's performance is said to be good. In case of mobile ad hoc networks, due to the limited resources, the nodes that have agreed to forward packets may not do so. Then such nodes are said to be misbehaving. If the algorithm does not handle the misbehaving nodes properly, then the efficiency of the algorithm degrades.

This work simulates a mobile ad hoc network, under the network simulator environment from UC Berkeley. The algorithm is run under the simulator, and its misbehavior has been studied for varying network size, bandwidth, traffic pattern, and mobility ratio and battery power. The performance of the algorithm is evaluated against end-to-end delay, delivery rate, overhead ratio and percentage bandwidth utilization.

In the implementation, Tcl script has been used to configure the network and to simulate it. A routine in C has been written to extract the simulation results from the output file generated by network simulator. The results of simulations have been shown by the graphs drawn in MS Excel.

CONTENTS

1. INTRODUCTION	1
1.1 Background	1
1.2 What is an Ad hoc Network	2
1.3 Characteristics of MANET	3
1.4 Routing in Mobile Ad hoc Networks	4
1.5 Applications of Ad hoc Networks	7
1.6 Problem Definition	8
1.7 Organization of the Dissertation	9
2. ROUTING MISBEHAVIOR	10
3. DYNAMIC SOURCE ROUTING	12
3.1 Introduction	12
3.2 Assumptions	14
3.3 Protocol Overview	14
3.3.1 Basic Route Discovery	14
3.3.2 Route Maintenance	16
3.4 Salient Features of the Algorithm	17
3.5 Conceptual Data Structures	20
4. SIMULATION TOOL USED & METHODOLOGY	24
4.1 The Simulator	24
4.2 Methodology	27
5. EXPERIMENTATION AND RESULTS	32

5.1 Simulation Experiment	32
5.2 Simulation Results	35
5.2.1 End-to-End Delay	36
5.2.2 Delivery Rate	38
5.2.3 Overhead Ratio	41
5.2.4 Percentage Bandwidth Utilization	43
CONCLUSION & FUTURE WORK	46
BIBLIOGRAPHY	47
Appendix A (DSR Packet Format)	
Appendix B (Simulation Results in Tabular format)	

CHAPTER 1
INTRODUCTION

Now-a-days people are using various portable devices in their day-to-day life. These devices can be a laptop, a palmtop, a mobile phone or a personal digital assistant (PDA). The mobile devices are not being used for various ranges of applications. People are using them as a means of communication only. Their applications do not interact with each other. If they could interact then a wide variety of applications can be seen like participants of a meeting exchanging documents, sharing data and as passengers exit a train, their laptops could remain online; likewise, incoming email could now be diverted to their PDAs. All these applications often can be referred to as *ad hoc* and the technology behind this is known as **Mobile Ad hoc NETWORK (MANET)**.

1.1 Background

In the present situation, the wireless data communication has gained lot of prominence among the users of the Internet, with many people using the mobile phones to access the Internet. The future looks to be very bright for the short range data communication devices, serving as a complement to traditional large scale communication, most man machine communication as well as oral communication between human beings occurs at distances of less than 10 meters also, as a result of this communication, the two communicating parties often have a need to exchange data. So there should be a technology that could govern this short range communication. The *mobile ad hoc networking* stands as such technology that will enable the short range devices to communicate with each other through exchanging documents and sharing data.

1.2 What is an ad hoc network?

The concept of passing along a message from one person to another is the concept underpinning the mobile ad hoc networks. If the term is defined basing on the meaning of the words in the term then the mobile ad hoc network can be defined as: network is a means for communication, ad hoc means there is no reliance on pre established hierarchy, and mobile implies non stationary elements. Thus *a mobile ad hoc network is a system for communicating between moving nodes whose configuration may change at any time.*

The vision of mobile networking is to support robust and efficient operation in mobile wireless networks by incorporating the routing functionality in to the mobile nodes. Such networks are envisioned to have dynamic, some times rapidly changing, random, multi hop topologies which are likely composed of relatively bandwidth constrained wireless links. Each node of a MANET consists of an electronic device capable of two things: bi-directional communication with nodes within range and routing of data.

Areas in which there is no communication infrastructure available or the existing infrastructure is expensive or inconvenient to use, wireless mobile users may still be able to communicate through the formation of an ad hoc network. An ad hoc network is an infrastructure less network. Infrastructure less networks do not have fixed infrastructure for the purpose of routing of data. All the nodes are capable of movement and can be connected dynamically in arbitrary manner. Nodes of these networks function as routers, which discover and maintain routes to other nodes in the network. Nodes are free to enter and leave the network at any time due to high mobility of the nodes. Because of the limited transmission range of the nodes, multiple hops may be needed to reach other nodes. Ad hoc networks are also capable of handling topological changes and malfunction in nodes. If a node leaves the network and causes the link breakages, affected nodes can easily request new routes and the problem will be solved.

1.3 Characteristics of MANET

MANET consists of mobile platforms, which are free to move about arbitrarily. MANET is an autonomous system of mobile nodes. The system may operate in isolation, or may have gateways to and interface with a fixed network. In the latter operational mode, it is typically envisioned to operate as a "stub" network connecting to a fixed Internetwork. Stub networks carry traffic originating at and/or destined for internal nodes, but do not permit exogenous traffic to "transit" through the stub network. MANET nodes are equipped with wireless transmitters and receivers using antennas, which may be omni directional (broadcast), highly-directional (point-to-point), possibly steerable, or some combination thereof. At a given point in time, depending on the nodes' positions and their transmitter and receiver coverage patterns, transmission power levels and co-channel interference levels, a wireless connectivity in the form of a random, multi hop graph or an ad hoc network exists between the nodes. This ad hoc topology may change with time as the nodes move or adjust their transmission and reception parameters. MANETs have the following salient characteristics:

- **Dynamic topologies:** Nodes are free to move arbitrarily; thus, the network topology which is typically multi hop may change randomly and rapidly at unpredictable times, and may consist of both bi directional and unidirectional links.

- **Bandwidth-constrained, variable capacity links:** Wireless links will continue to have significantly lower capacity than their hardwired counterparts. In addition, the realized throughput of wireless communications after accounting for the effects of multiple access, fading, noise, and interference conditions, etc. is often much less than a radio's maximum transmission rate. One effect of the relatively low to moderate link capacities is that congestion is typically the norm rather than the exception, i.e. aggregate application demand will likely approach or

exceed network capacity frequently. As the mobile network is often simply an extension of the fixed network infrastructure, mobile ad hoc users will demand similar services. These demands will continue to increase as multimedia computing and collaborative networking applications rise.

- **Energy-constrained operation:** Some or all of the nodes in a MANET may rely on batteries or other exhaustible means for their energy. For these nodes, the most important system design criteria for optimization may be energy conservation.

- **Limited physical security:** Mobile wireless networks are generally more prone to physical security threats than are fixed-cable nets. The increased possibility of eavesdropping, spoofing, and denial-of-service attacks should be carefully considered. Existing link security techniques are often applied within wireless networks to reduce security threats. As a benefit, the decentralized nature of network control in MANETs provides additional robustness against the single points of failure of more centralized approaches [5],[7].

1.4 Routing In Mobile Ad hoc Networks

For *mobile ad hoc* networks, the issue of routing packets between any pair of nodes becomes a challenging task because the nodes can move randomly within the network. A path that was considered optimal at a given point in time might not work at all a few moments later. Traditional routing protocols are *proactive* in that they maintain routes to all nodes, including nodes to which no packets are being sent. They react to any change in the topology even if no traffic is affected by the change, and they require periodic control messages to maintain routes to every node in the network. The rate at which these control messages are sent must reflect the dynamics of the network in order to maintain valid routes. Thus, scarce

resources such as power and link bandwidth will be used more frequently for control traffic as node mobility increases. An alternative approach involves establishing *reactive* routes, which dictates that routes between nodes are determined solely when they are explicitly needed to route packets. This prevents the nodes from updating every possible route in the network, and instead allows them to focus either on routes that are being used, or on routes that are in the process of being set up. The proactive approach depletes too many resources updating paths (if the route update periods are to match the mobility of the nodes). If the update interval is too long, the network will simply contain a large amount of stale routes in the nodes, which results in a significant loss of packets. Another approach, which is both proactive and reactive, is called as *Hybrid* approach.

Some of the most prominent routing protocols for mobile ad hoc networks are discussed below.

Destination Sequenced Distance Vector (DSDV)

In this protocol every node is assigned a monotonically increasing, even sequence number. Routes are maintained at each node and broadcast to neighboring nodes. The routing information consists of the sequence number of the next hop in the chain, the number of hops to the destination and the destination's address. Packets are routed using the sequence numbers in such a way that looping is prevented. The packet travels along the highest numbered path with number of hops being equal and minimized [3].

Dynamic Source Routing (DSR)

DSR nodes contain a cache of full routes paths between node pairs. To discover a route, a Route Request packet and is broadcast and flooded to all nodes. The destination node knowing a route to the destination replies with a Route reply packet that is also propagated throughout the network. Each node hearing a Route

Reply message stores the entire path to the destination in its cache. The protocol is thus divided into two phases. Route Discovery and Route maintenance. Data packets are transmitted with the entire route stored in the packet itself [6].

Ad Hoc On-Demand Distance Vector Routing (AODV)

AODV works by combining the features of both DSR and DSDV. As in DSR routes are discovered by controlled propagation of Route Request and Route Reply packets. Each node only keeps track of the next hop towards a particular destination and utilizes sequence numbers to avoid looping. Periodic Hello messages are broadcast once per second to maintain the link status. Three failed messages indicate that a link is down. Alternatively, physical layer or link layer methods may be used to keep track of neighbors. Unsolicited Route Reply messages are generated when links go down to reset routing information [2].

Temporally Ordered Routing Algorithm (TORA)

This protocol attaches a 'height' value that decreases linearly over each hop to a destination, the height is basically the number of hops. Searchers find routes by broadcasting a Query packet to all neighbors. Each neighbor relays this packet to all of its neighbors until the destination, or a node knowing the height to the destination, is found. The final node then broadcasts an update packet that contains the height and this packet propagates through out the network, with each node updating its height accordingly. A Clear packet is used to reset the routing data when a node becomes unreachable.

As far as the algorithms behave the way they are expected to, then the throughput of the ad hoc network will be very good. When a node agrees to forward a packet but fails to do so, then it can be said that the node is misbehaving. Once the nodes start misbehaving then the throughput of the algorithm declines. Packets are not forwarded through these nodes and if these nodes are intermediate nodes then the

packets that are being forward from a source to a destination via these intermediate nodes may also be dropped [16].

The ad hoc routing algorithms can be classified as shown below.

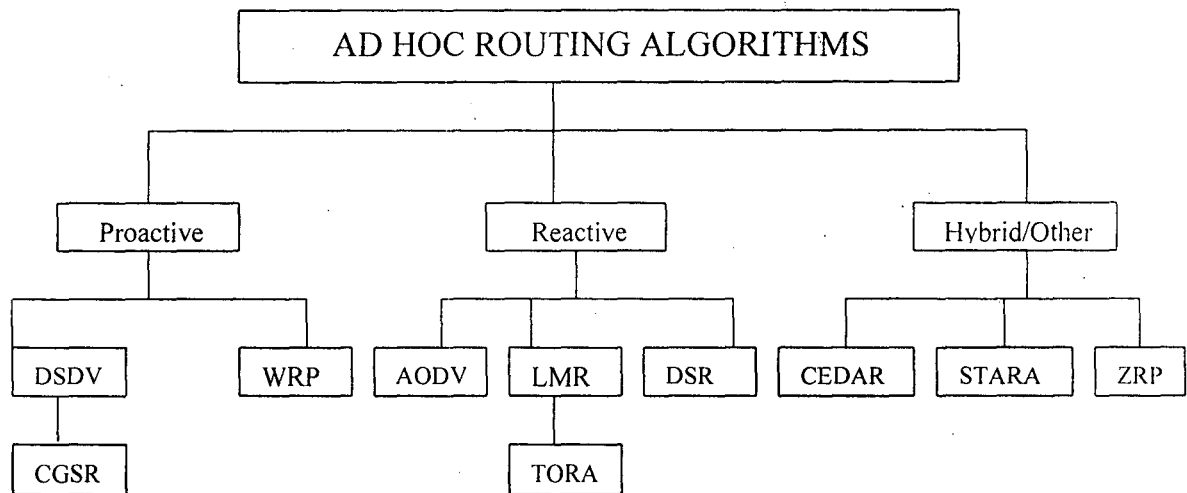


Fig 1.1 The various Routing Protocols and their relation

1.5 Applications of Ad Hoc Networks

There is current and future need for dynamic ad hoc networking technology. The emerging field of mobile and nomadic computing, with its current emphasis on mobile IP operation, should gradually broaden and require highly-adaptive mobile networking technology to effectively manage multi hop, ad hoc network clusters which can operate autonomously or, more than likely, be attached at some point(s) to the fixed Internet.

Some applications of MANET technology could include industrial and commercial applications involving cooperative mobile data exchange. In addition, mesh-based mobile networks can be operated as robust, inexpensive alternatives

or enhancements to cell-based mobile network infrastructures. There are also existing and future military networking requirements for robust, IP-compliant data services within mobile wireless communication networks, many of these networks consist of highly-dynamic autonomous topology segments. Also, the developing technologies of "wearable" computing and communication say provide applications for MANET technology. When properly combined with satellite-based information delivery, MANET technology can provide an extremely flexible method for establishing communications for fire/safety/rescue operations or other scenarios requiring rapidly deployable communications with survivable, efficient dynamic networking [7].

1.6 Problem Definition

As described in the earlier section 1.4 that a node may misbehave by not forwarding the data packets having agreed to forward the data packets. This characteristic of a node has significant impact on the performance of the network. The impact on the performance may vary depending on the underlying routing algorithm used by the network.

In view of the above, the present study focuses on routing misbehavior of DSR algorithm in context of following network parameters.

- Network size
- Bandwidth
- Traffic pattern
- Mobility rate
- Battery power

In the situation of routing misbehavior the performance of the algorithm will be evaluated. The following are the parameters for evaluation.

- End to End Delay
- Delivery Rate
- Overhead Ratio
- Percentage Bandwidth Utilization

The effect of routing misbehavior on the performance of the algorithm is to be studied by varying one or more of these parameters.

1.7 Organization of the Dissertation

The present chapter gives the basic introduction to the mobile ad hoc network, different routing protocols used in MANET and various applications of MANET. Chapter2 describes the concept of routing misbehavior. The details of Dynamic source routing algorithm are discussed in chapter 3. Chapter 4 gives the methodology, tools and techniques used, in the present work. The experiments, results obtained from the experiments and their analysis have been described in Chapter 5. Finally the thesis summarizes with the conclusion of the present work and the scope for the future enhancements, followed by a well-referenced bibliography and three appendices providing header formats, simulation results in tabular format and abbreviations of the terms used in the thesis.

ROUTING MISBEHAVIOR

Ad hoc networks maximize total network throughput by using all available nodes for routing and forwarding. Therefore, the more nodes that participate in packet routing, the greater the aggregate bandwidth, the shorter the possible routing paths, and the smaller the possibility of a network partition. However, a node may misbehave by agreeing to forward packets and then failing to do so, because it is overloaded, selfish, malicious, or broken.

- An overloaded node lacks the CPU cycles, buffer space or available network bandwidth to forward packets.
- A selfish node is unwilling to spend battery life, CPU cycles, or available network bandwidth to forward packets not of direct interest to it, even though it expects others to forward packets on its behalf.
- A malicious node launches a denial of service attack by dropping packets.
- A broken node might have a software fault that prevents it from forwarding packets.

Misbehaving nodes can be a significant problem. Even a few misbehaving nodes can have a severe impact on intermediate nodes.

One solution to misbehaving nodes is to forward packets only through nodes that share an a priori trust relationship. A priori trust relationships are based on pre-existing relationships built outside of the context of the network (e.g. friendships, companies, and armies). The problems with relying on a priori trust-based forwarding are that

- it requires key distribution,
- trusted nodes may still be overloaded,
- trusted nodes may still be broken,

- trusted nodes may be compromised, and
- untrusted nodes may be well behaved.

It may not be possible to exchange keys used to authenticate trusted nodes outside of the ad hoc network before the conference or disaster that requires an ad hoc network. If keys are not distributed ahead of time, then enforcing a priori trust-based forwarding requires a secure channel for key exchanges within the ad hoc network for authentication. Even if keys can be exchanged, a trusted node's security may be compromised, or a trusted node may be overloaded or broken as mentioned above. Finally, although relying on a priori trust-based forwarding reduces the number of misbehaving nodes, it also excludes untrusted well behaved nodes whose presence could improve ad hoc network performance. Another solution to misbehaving nodes is to attempt to forestall or isolate these nodes from within the actual routing protocol for the network. However, this would add significant complexity to protocols whose behavior must be very well defined. In fact, current versions of mature ad hoc routing algorithms, including DSR, AODV, TORA, DSDV, STAR, and others only detect if the receiver's network interface is accepting packets, but they otherwise assume that routing nodes do not misbehave. Although trusting all nodes to be well behaved increases the number of nodes available for routing, it also admits misbehaving nodes to the network [15].

DYNAMIC SOURCE ROUTING ALGORITHM

3.1 Introduction

The dynamic source routing algorithm (DSR), is a simple and efficient routing protocol designed specifically for use in multi hop wireless ad hoc networks of mobile nodes. DSR allows the network to be completely self-organizing and self-configuring, with out the need for any existing infrastructure or administration. Nodes in the network cooperate to forward packets to each other to allow communication over the multiple hops, which are not in the wireless transmission range of each other. As the nodes in the network can move anywhere, can join or leave the network at any time and the as wireless transmission conditions like sources of interference change, all routing is automatically determined and is maintained by the DSR protocol. As the number or sequence of intermediate hops needed to reach any destination may change at any time, the resulting network is highly dynamic in nature and is rapidly changing.

In DSR algorithm every data packet sent contains in its header the complete ordered list of nodes through which packet will pass. This type of routing is known as source routing. The DSR protocol enables the nodes to discover a source route dynamically to any destination across multiple hops or across multiple network hops. Hence, the name dynamic source routing algorithm. This algorithm allows the packet routing to be loop- free and avoids the need for up-to-date routing information in the intermediate nodes through which the packet is being forwarded. By appending the source route in the packet header, the other nodes forwarding or overhearing any of these data packets may also easily cache this routing information for future use.

The protocol is composed of the two mechanisms *Route Discovery* and *Route Maintenance*, which allow nodes to discover and maintain source routes to arbitrary destination in the ad hoc network. All aspects of the protocol operate entirely on-demand, allowing the routing packet overheads of DSR to scale automatically to only those nodes that need to react to the changes in the routes currently in use.

Route Discovery is the mechanism by which a node **A** willing to send a packet to a destination node **G**, obtains the path to **G**. Route discovery is used only when **A** wants to send a packet to node **G** but does not know a path to **G**.

Route Maintenance is the mechanism through which a node **A** is able to detect that it can no longer use its route to **G** because a link along the route no longer exists due to a change in network topology. When Route Maintenance indicates a source route is broken, **A** can attempt to use any other route it happens to know to **G**, or can invoke Route Discovery again to find a new route for subsequent packets. Route Maintenance for this route is used only when **A** is actually sending packets to **G**.

In DSR algorithm, Route Discovery and Route Maintenance each operate entirely *on demand*. DSR algorithm requires no periodic packets of any kind at any level within the network. For example, DSR algorithm does not use any periodic routing advertisement, link status sensing, or neighbor detection packets, and does not rely on these functions from any underlying protocols in the network. This entirely on-demand behavior and lack of periodic activity allows the number of overhead packets caused by DSR to scale all the way down to zero, when all nodes are approximately stationary with respect to each other and all routes needed for current communication have already been discovered. As nodes begin to move more or as communication patterns change, the routing packet overhead of DSR automatically scales to only that needed to track the routes currently in use. Network topology changes not affecting routes currently in use are ignored and do not cause reaction from the protocol. In response to a single Route

Discovery (as well as through routing information from other packets overheard), a node may learn and cache multiple routes to any destination. This allows the reaction to routing changes to be much more rapid, since a node with multiple routes to a destination can try another cached route if the one it has been using fails. This caching of multiple routes also avoids the overhead of needing to perform a new Route Discovery each time a route in use breaks. The operation of both Route Discovery and Route Maintenance in DSR are designed to allow unidirectional links and asymmetric routes to be easily supported. In wireless networks, it is possible that a link between two nodes may not work equally well in both directions, due to differing antenna or propagation patterns or sources of interference. DSR allows such unidirectional links to be used when necessary, improving overall performance and network connectivity in the system.

3.2 Assumptions

All the hosts willing to communicate with other host in the ad hoc network, by using the DSR algorithm are assumed to participate fully in the protocols of the network. Each host participating in the network should also be willing to forward packets to the other nodes in the network.

The number of hops required for a packet to reach from one host located at the one extreme edge of the network to the opposite extreme is known as the diameter of the network.

Hosts within the ad hoc network may move at any time without notice but we assume that the speed with which hosts move is moderate with respect to the packet transmission latency and wireless transmission range of the particular underlying network hardware in use. We assume that hosts do not continuously move so rapidly

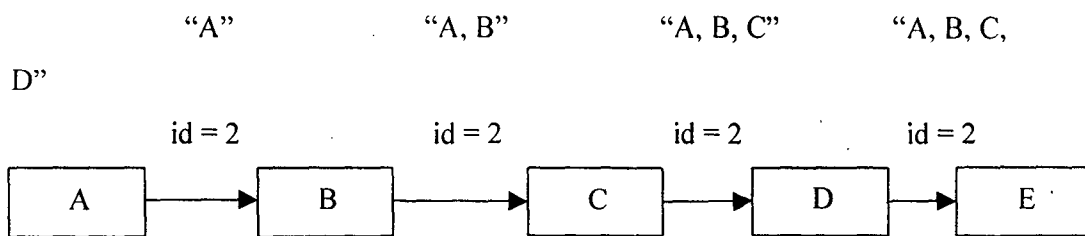
as to make the flooding of each packet the only possible routing protocol.

3.3 Protocol Overview

3.3.1 Basic Route Discovery:

When a source node originates a packet to a destination node, the source node places in the header of the packet a source route giving the sequence of hops that the packet has to follow in its way to the destination. The sender will obtain a suitable source route by searching its *Route Cache* of routes previously learned, but if no route is found in its cache for the intended destination then it initiates a route discovery to find the route to the destination dynamically. The source node is called the initiator and the destination node is called as the target of the route discovery.

For example let us consider the following scenario node A wants to initiate a route discovery to node E. The route discovery initiated by node A in this scenario will be as follows.



To initiate the route discovery the node A transmits a *route request* broadcast packet to all the nodes within its wireless range. Every route request identifies the initiator and target of the route request discovery, and also contains a unique identification (in this example it is 2) determined by the initiator of the request.

Each route request also contains a record listing the address of each intermediate node through which the route request packet has been propagated. The route Request initially lists only the node A. When another node receives the Route Request, if it is the target of route discovery, it returns a Route Reply to the node that initiated the Route Discovery. It also gives a copy of the accumulated route record from the Route Request. When the initiator receives the Route Reply it caches this route in its route cache for use in sending subsequent packets to this destination.

If this node receiving the Route Request has recently seen another Route Request message from this initiator bearing the same request identification and target, or if this node's own address is already listed in the route record in the Route Request, this node discards the request. This node appends its own address to the route record in the Route Request and propagates it by transmitting it as a local broadcast packet (with the same request identification).

When initiating the Route Discovery, the sending node saves a copy of the original packet in a local buffer called the *send buffer*. The send buffer contains a copy of each packet that cannot be transmitted by this node because it does not yet have a source route to the packet's destination. Each packet in the send buffer is logically associated with the time it was placed in to the buffer. The packet is discarded if the packet still resides in the send buffer for that time period.

3.3.2 Route Maintenance

When originating or forwarding a packet using a source route, each node transmitting the packet is responsible for confirming that the packet should be retransmitted until this confirmation of receipt is received. This confirmation can be obtained as a MAC layer acknowledgement or as a passive acknowledgement. If none of the two confirmations are received the node transmitting the packet can explicitly request a DSR-specific software acknowledgement can be returned by the next hop. This software acknowledgement can normally be transmitted

directly to the sending node. If the link between the two nodes is unidirectional then the software acknowledgement may take a different multi hop path.

If no receipt information is received after the packet has been retransmitted the maximum no. of attempts by the same hop, this node should return a Route Error packet to the original sender of the packet, identifying the link on which the packet could not be forwarded

3.4 Salient Features of the Algorithm

➤ Caching of Overhead Routing Information

A node forwarding or overhearing any packet may add the routing information from that packet to its own *Route Cache*. In particular, the source route used in a data packet, any node may cache all the accumulated route record in a Route request, or on the route being returned in a Route Reply. Routing information from any of these packets received can be cached, whether the packet was addressed to this node, sent to a broad cast MAC address, or received while the node's network interface is in promiscuous mode.

➤ Replying to a Route Requests using Cached Routes

A node receiving a Route request for which it is not the target searches in its route cache for a route to the target of the target of the Request. If found, the node generally returns a Route Reply to the initiator itself rather than forwarding the Route Request. In the Route Reply, this node sets the route record to list the sequence of hops over which this copy of the Route Request was forwarded to it, concatenated with the source route to this target obtained from its own Route Cache. However, before transmitting a Route Reply packet that was generated using information from its Route Cache in this way, a node must verify that the resulting route being

returned in the Route Reply, after this concatenation, contains no duplicate nodes listed in the route record.

➤ **Preventing Route Reply Storms**

The ability for nodes to reply to a Route Request based on information in their Route Caches could result in a possible Route Reply "storm" in some cases. In particular, if a node broadcasts a Route Request for a target node for which the node's neighbors have a route in their Route Caches, each neighbor may attempt to send a Route Reply, thereby wasting bandwidth and possibly increasing the number of network collisions in the area.

Normally, these nodes would all attempt to reply from their own Route Caches, and would all send their Route Replies at about the same time, since they all received the broadcast Route Request at about the same time. Such simultaneous replies from different nodes all receiving the Route Request may create packet collisions among some or all of these Replies and may cause local congestion in the wireless network. In addition, it will often be the case that the different replies will indicate routes of different lengths.

➤ **Route Request Hop limits**

Each Route Request message contains a "hop limit" that may be used to limit the number of intermediate nodes allowed to forward that copy of the Route Request. This hop limit is implemented using the Time-to-Live (TTL) field in the IP header of the packet carrying the Route Request. As the Request is forwarded, this limit is decremented, and the Request packet is discarded if the limit reaches zero before finding the target.

This Route Request hop limit can be used to control the spreading of a Route Request during a Route Discovery attempt. For example, a node may send its first Route Request attempt for some target node using a hop limit of 1, such that any node receiving the initial transmission of the Route Request will not forward the Request to other nodes by rebroadcasting it. This form of Route Request is called a "non-propagating" Route Request. It provides an inexpensive method for determining if the target is currently a neighbor of the initiator or if a neighbor node has a route to the target cached.

➤ **Packet Salvaging**

After sending a Route Error message as part of Route Maintenance, a node may attempt to salvage the data packet that caused the Route Error rather than discarding the packet. To attempt to salvage a packet, the node sending a Route Error searches its own Route Cache for a route from itself to the destination of the packet causing the Error. If such a route is found, the node may salvage the packet after returning the Route Error by replacing the original source route on the packet with the route from its Route Cache. The node then forwards the packet to the next node indicated along this source route. When salvaging a packet in this way, a count is maintained in the packet of the number of times that it has been salvaged, to prevent a single packet from being salvaged endlessly. Otherwise, it could be possible for the packet to enter a routing loop, as different nodes repeatedly salvage the packet and replace the source route on the packet with routes to each other.

➤ **Automatic Route Shortening**

Source routes in use may be automatically shortened if one or more intermediate hops in the route become no longer necessary. This mechanism of automatically shortening routes in use is

somewhat similar to the use of passive acknowledgements. In particular, if a node is able to overhear a packet carrying a source route (e.g., by operating its network interface in promiscuous receive mode), then this node examines the unused portion of that source route. If this node is not the intended next hop for the packet but is named in the later unused portion of the packet's source route, then it can infer that the intermediate nodes before itself in the source route are no longer needed in the route.

➤ **Increased Spreading of Route Error Messages**

When a source node receives a Route Error for a data packet that it originated, this source node propagates this Route Error to its neighbors by piggybacking it on its next Route Request. In this way, stale information in the caches of nodes around this source node will not generate Route Replies that contain the same invalid link for which this source node received the Route Error. On the Route Request packet initiating this Route Discovery, node A piggybacks a copy of this Route Error, ensuring that the Route Error spreads well to other nodes, and guaranteeing that any Route Reply that it receives (including those from other node's Route Caches) in response to this Route Request does not contain a route that assumes the existence of this broken link.

3.5 Conceptual Data Structures

DSR algorithm uses the following data structures for transferring data across a mobile ad hoc network.

➤ **Route Cache**

All routing information needed by a node participating in an ad hoc network using DSR is stored in that node's Route Cache. Each node in the

TH-9441

network maintains its own Route Cache. A node adds information to its Route Cache as it learns of new links between nodes in the ad hoc network, for example, a node may learn of new links when it receives a packet carrying either a Route Reply or a DSR Routing header. Likewise, a node removes information from its Route Cache as it learns that existing links in the ad hoc network have broken for example, a node may learn of a broken link when it receives a packet carrying a Route Error or through the link-layer retransmission mechanism reporting a failure in forwarding a packet to its next-hop destination. It is possible to interface a DSR network with other networks, external to this DSR network. Such external network may be the Internet, or may be other ad hoc networks routed with a routing protocol other than DSR. Such external networks may also be other DSR networks that are treated as external networks in order to improve scalability. This minimal set of requirements and features involve the First Hop External (F) and Last Hop External (L) bits in a Source Route option (Section 5.7) and a Route Reply option (Section 5.3) in a packet's DSR header (Section 5). These requirements also include the addition of an External flag bit tagging each node in the Route Cache, copied from the First Hop External (F) and Last Hop External (L) bits in the Source Route option or Route Reply option from which the link to this node was learned. The Route Cache should support storing more than one route to each destination. In searching the Route Cache for a route to some destination node, the Route Cache is indexed by destination node address. An implementation of a Route Cache may provide a fixed capacity for the cache, or the cache size can be a variable.

➤ **Route Request Table**

The Route Request Table records information about Route Requests that have been recently originated or forwarded by this node. The table is indexed by IP address. The Route Request Table on a node records the



following information about nodes to which this node has initiated a Route Request:

- The time that this node last originated a Route Request for that target node.
- The number of consecutive Route Requests initiated for this target since receiving a valid Route Reply giving a route to that target node
- The remaining amount of time before which this node may next attempt at a Route Discovery for that target node.
- The Time-to-Live (TTL) field used in the IP header of last Route Request initiated by this node for that target node.

In addition, the Route Request Table on a node also records the following information about initiator nodes from which this node has received a Route Request

A FIFO cache of size REQUEST_TABLE_IDS entries containing the Identification value and target address from the most recent Route Requests received by this node from that initiator node.

Nodes should use an LRU policy to manage the entries in their Route Request Table. The number of Identification values to retain in each Route Request Table entry, REQUEST_TABLE_IDS, must not be unlimited, since, in the worst case, when a node crashes and reboots, the first REQUEST_TABLE_IDS Route Discoveries it initiates after rebooting could appear to be duplicates to the other nodes in the network. In addition, a node should base its initial Identification value, used for Route Discoveries after rebooting, on a battery backed-up clock or other persistent memory device, in order to help avoid any possible such delay in successfully discovering new routes after rebooting if no such source of initial Identification value is available, a node should base its initial Identification value after rebooting on a random number.

➤ **Send Buffer**

The Send Buffer of a node implementing DSR is a queue of packets that cannot be sent by that node because it does not yet have a source route to each such packet's destination. Each packet in the Send Buffer is logically associated with the time that it was placed into the Buffer, and should be removed from the Send Buffer and silently discarded SEND_BUFFER_TIMEOUT seconds after initially being placed in the Buffer. If necessary, a FIFO strategy can be used to evict packets before they timeout to prevent the buffer from overflowing. A Route Discovery should be initiated as often as possible for the destination address of any packets residing in the Send Buffer.

➤ **Retransmission Buffer**

The Retransmission Buffer of a node implementing DSR is a queue of packets sent by this node that are awaiting the receipt of an acknowledgment from the next hop in the source route. For each packet in the Retransmission Buffer, a node maintains

- A count of the number of retransmissions and
- The time of the last retransmission

Packets are removed from the Retransmission Buffer when an acknowledgment is received or when the number of retransmissions exceeds DSR_MAXRXTSHIFT. In the later case, the removal of the packet from the Retransmission Buffer should result in a Route Error being returned to the original source of the packet [6]

SIMULATION TOOL USED & METHODOLOGY

In this dissertation work, a study of the effect of routing misbehavior on DSR algorithm is made. For studying the misbehavior a tool called network simulator from UC Berkeley is used. Network Simulator (ns2) is written in C++ and OTcl. NS 2 is being widely used for simulating the local area networks, wide area network and wireless network. Ad hoc networks are also supported by NS2.

4.1 The Simulator

NS uses two languages because simulator has two different kinds of things it needs to do. The detailed simulations of protocols requires a systems programming language which can efficiently manipulate bytes, packet headers, and implement algorithms that run over large data sets. For these tasks run-time speed is important and turn around time is less important. On the other hand there is a need for varying parameters or configurations, or quickly exploring a number of scenarios. Since configuration runs once, runtime of this part of the task is less important. NS meets both of these needs with two languages, OTcl and C++. C++ is fast enough to run but slower to change, making it suitable for detailed implementation. OTcl runs much slower but can be changed very quickly, making it ideal for simulation configuration. NS provides glue to make variables and objects appear on both the languages [11].

NS is an event driven network simulator developed at UC Berkeley that simulates wide variety of IP networks. It implements network protocols such as TCP and UDP, traffic source behavior such as FTP, Telnet, CBR and VBR, router queue management mechanisms like Drop Tail, RED, CBQ, routing algorithms such as Dijkstra, and more. NS also implements multicasting and some of the MAC layer protocols for LAN simulations. The current version NS2 is written in C++ and

OTcl. OTcl is an object oriented extension of the scripting language Tcl and is developed by MIT.

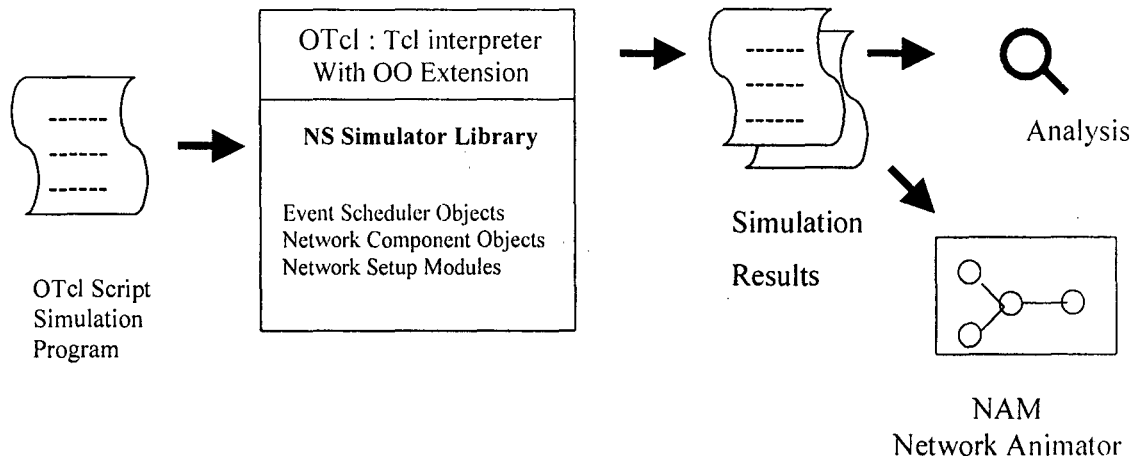


Fig. 4.1 Simplified User view OF NS

NS is an object oriented Tcl (OTcl) script interpreter that has a simulation event scheduler and network component object libraries, and network setup module libraries. To start simulation using NS, a OTcl script should be written which initiates the event scheduler and sets up the network topology using the network objects and the plumbing functions in the library, and tells traffic sources when to start and stop transmitting packets through the event scheduler. A network object can easily be created by writing a new object or by making the a compound object from the object library. The power of NS comes from how it creates the network objects and the network components.

Another major component of NS beside network objects is the event scheduler. An event in NS is a packet ID that is unique for a packet with scheduled time and the pointer to an object that handles the event. In NS, an event scheduler keeps track of simulation time and fires all the events in the event queue scheduled for the current time by invoking appropriate network components, which usually are

the ones who issued the events, and let them do the appropriate action associated with packet pointed by the event. Network components communicate with one another passing packets, however this does not consume actual simulation time. All the network components that need to spend some simulation time handling a packet (i.e. need a delay) use the event scheduler by issuing an event for the packet and waiting for the event to be fired to itself before doing further action handling the packet. For example, a network switch component that simulates a switch with 20 microseconds of switching delay issues an event for a packet to be switched to the scheduler as an event 20 microseconds later. The scheduler after 20 microseconds dequeues the event and fires it to the switch component, which then passes the packet to an appropriate output link component. Another use of an event scheduler is timer. For example, TCP needs a timer to keep track of a packet transmission time out for retransmission (transmission of a packet with the same TCP packet number but different NS packet ID). Timers use event schedulers in a similar manner that delay does. The only difference is that timer measures a time value associated with a packet and does an appropriate action related to that packet after a certain time goes by, and does not simulate a delay. NS is written not only in OTcl but in C++ also. For efficiency reason, NS separates the data path implementation from control path implementations. In order to reduce packet and event processing time (not simulation time), the event scheduler and the basic network component objects in the data path are written and compiled using C++. These compiled objects are made available to the OTcl interpreter through an OTcl linkage that creates a matching OTcl object for each of the C++ objects and makes the control functions and the configurable variables specified by the C++ object act as member functions and member variables of the corresponding OTcl object. In this way, the controls of the C++ objects are given to OTcl. It is also possible to add member functions and variables to a C++ linked OTcl object. The objects in C++ that do not need to be controlled in a simulation or internally used by another object do not need to be linked to OTcl. Likewise, an object (not in the data path) can be entirely implemented in OTcl. Figure 2 shows an object hierarchy example in C++ and OTcl. One thing to note in the figure is that for C++ objects that have an

OTcl linkage forming a hierarchy, there is a matching OTcl object hierarchy very similar to that of C++.

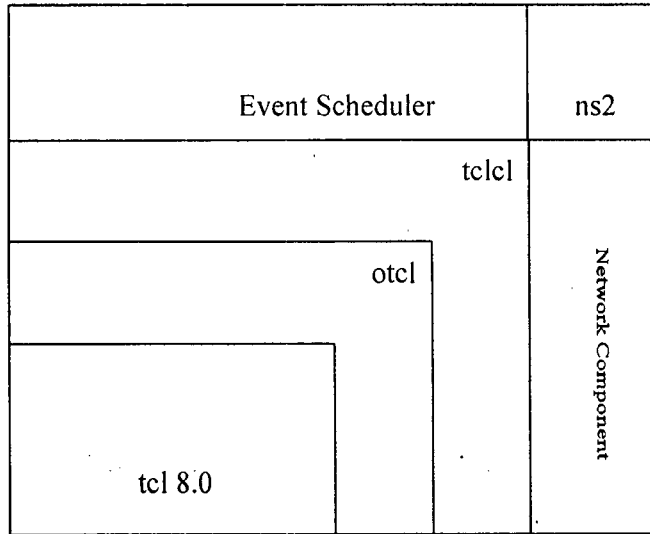


Fig 4.2 Architecture Of NS2

In the above figure the user view of the NS2 has been given. If it is assumed that the user is at the bottom left corner he will design and run simulations written in Tcl using the OTcl library. The user components and most of the network components are written in C++ and are available in OTcl through the OTcl linkage that is implemented using Tclcl. The entire thing together makes NS.

When a simulation is run under the NS2 environment, NS produces one or more text-based output files that contain the detailed simulation data, if specified to do so in the Tcl script. The data can be used for simulation analysis or as an input to a graphical simulation display tool called network animator (NAM) [8] .

4.2 Methodology

In this study an ad hoc network that works on dynamic source routing algorithm has been simulated using the ns2. For creating a mobile node NS2 provides with the class Mobile node. The characteristics of the mobile node can be set using the *config()* function defined on the mobile node. The parameters like routing protocol being used, the link layer protocol on which the algorithm works, battery power used for transmission of data and for receiving of data etc, can be set using this class. Whenever a node is created in NS using this mobile node class, all the parameters are set on the node. There after the nodes behave as per their specifications.

As ad hoc networks are highly mobile in nature, there should be a support for the high mobility of the nodes. The movement of the nodes can be configured in such a way that the network is highly mobile and ad hoc in nature. NS is designed in such a way that the nodes can enter and leave the network at any time. For giving the dynamic nature to the algorithm ns makes use of two files called mobility file and traffic pattern file. The network simulator uses the mobility file to induce the node movement in to the network and the traffic pattern file is used to introduce the traffic across the various wireless nodes.

The rate with which the nodes move can be specified in the mobility file and the traffic pattern can be specified in the traffic pattern file like bursty traffic or normal traffic by specifying the number of connections across the network nodes. Like wise the battery power, the initial energy of the cells in the mobile nodes can be fixed in the NS2 by using the energy model. The power that was consummated when transferring data packet out of the mobile node and the power that was consummated when packet is received by the mobile node can also be fixed using the energy model.

In the process of studying the effect of routing misbehavior of Dynamic Source Routing algorithm, a mobile ad hoc network has been simulated. The NS2 developed by UC Berkeley, expects that the nodes in the network behave the way they are expected to, based on the algorithm on which the network works. But, the concept of routing misbehavior of the nodes was not introduced in to the network simulator. To study the effect of the routing misbehavior on DSR algorithm, the NS2 needs to have a support for misbehaving nodes. For this purpose, a node misbehaving extension is made to the network simulator.

A variable is set on the node, which will characterize the node as a misbehaving node or not. This is set in the Tcl script, which is an input to the network simulator. Based on the value set on the parameter, the source code of DSR algorithm has to be modified in such a way that the nodes of the network act according to the value set on the variable.

The misbehaving nodes are selected at random using the random number generation facility in NS. Depending on the value set for the percentage of misbehaving nodes, random numbers are generated that will correspond to the node identifiers. In a network of m nodes if p percent nodes misbehave and this percentage computes to n , (i.e. the no of misbehaving nodes is n) then those n nodes out of m nodes are selected randomly. The data packets that are forwarded through them are dropped.

Once the misbehavior functionality has been induced on the nodes in the network, then the algorithm can be studied in the context of misbehaving nodes by varying the one or all of the parameters - network size, bandwidth, traffic pattern, Mobility rate and Battery Power.

The way the parameters are varied is discussed below.

- **Network Size** : As the network size increases and the percentage of misbehaving nodes varies, the size of the network will have its effect on

the misbehavior of the algorithm. The performance of the algorithm is to be tested for a network size of 25, 50, 75 and 100 nodes. Each time the percentage of misbehaving nodes is varied from 10 % to 50% with an increment of 10%.

- **Bandwidth** : The amount of bandwidth the network can support will have a lot of impact on the performance of the algorithm. The efficiency of the algorithm varies when the bandwidth of the network varies. So, the performance of the algorithm is tested for a network of bandwidth 1Mbps, 2Mbps, 5.5Mbps and 11Mbps.
- **Mobility Rate** : Mobility rate is the rate with which the mobile nodes move in the network. If the mobility rate is very high then it could lead to a partition in the network. If the network is broken then the intermediate nodes have to wait for a certain amount time to forward packets from one broken network to another, until there exists a link between the two broken networks. This may lead to the increased delay in the network. Hence the mobility rate should be the optimal. So the performance of the algorithm is tested for mobility rates of 5m/s, 10 m/s, 25 m/s and 50 m/s.
- **Traffic Pattern** : The traffic pattern that a network follows has a lot of impact on the network. If there are large number of nodes and there is very less traffic (normal traffic) then some nodes may not participate in routing, and on the other way, if there is very bursty traffic then the network may be congested. So, the optimal traffic pattern should be used. This traffic pattern may affect the end-to-end delay.
- **Battery Power** : Battery power is one of the resource, that has to be used very care fully in the context of mobile networks, this is being a limited resource. As some energy is consumed for transmitting and receiving packets, it may some times happen that some intermediate nodes may run out of battery power due to the over forwarding of data packets, or may not be able to forward data packets due to its limited battery power. In

such a situation, the battery power is also to be considered as an important factor because a node running out of power may drop a packet, which is being forwarded through it. This may result in loss of packets and will have its impact on the performance of the algorithm. In this experimentation, the power of the battery has been varied from 10J, 20J, 30 J, 40 J and 50 J, and the performance is tested.

Once the algorithm has been simulated and the results of the simulation are obtained, then the performance of the algorithm has to be evaluated on the parameters defined below.

- **End-to-End Delay** : This is also referred to as latency, and is the time needed to delivery the message. Data delay can be divided into queuing delay and propagation delay. If queuing delay is ignored, propagation delay can be replaced by hop count, because of proportionality. Retransmissions can be included if MAC (medium access control) layer is used in experiments.
- **Delivery Rate** : Delivery rate is defined as the ratio of numbers of messages received by destination to the no of messages sent by the senders per unit time.
- **Overhead Ratio** : It can be defined as the ratio of the total control bits transmitted to the actual data bits transmitted.
- **Percentage Bandwidth Utilization** : The percentage bandwidth utilization can be defined as the ratio of the bandwidth utilized in unit time across nodes of a network to the total bandwidth of the network.

EXPERIMENTATION & RESULTS

The previous chapter described the methodology for performing the simulation and discussed the basic steps involved in the process of simulating the network, under the network simulator environment. The current chapter discusses how the simulation is done and the simulation results are analyzed.

Simulations are carried out on a system running Pentium III processor with 1GHz speed, with a RAM of 256 MB, running Linux Operating System.

5.1 Simulation Experiment

The methodology followed in the process of running the simulator is explained in chapter4. In the process of running the simulator first a Tcl script is written, which is input to the NS and initiates the process of simulation. The script is written for creating the nodes, establishing the links, specifying the traffic patterns, specifying the node movement scenarios etc., energy model is also induced in to the network, which handles the battery power. The power with which the battery can transmit and receive the signals can be varied using the EnergyModel class defined in NS2.

In the present work, the misbehavior on the nodes was not introduced the way as it was described in Chapter4 due to the time constraints. A loss model was implemented, which will drop the packets, at specified percentage of drop like 10%, 20% etc, uniformly distributed over the nodes. In NS the loss model was implemented on the node. So, the percentage drop of packets is due to the misbehavior of the nodes in the network. In this way the misbehaving nodes concept has been induced in to the network, different from the way it was stated in

Chapter4. But, this implementation may be considered as an alternative of the method specified in Chapter4.

The sample Tcl script is shown below.

```
##### TO SET THE PARAMETERS FOR THE NETWORK #####
```

```
set val(chan) Channel/WirelessChannel
set val(prop) Propagation/TwoRayGround
set val(ant) Antenna/OmniAntenna
set val(ll) LL
set val(ifq) Queue/DropTail/PriQueue
set val(ifqlen) 50
set val(netif) Phy/WirelessPhy
set val(mac) Mac/802_11
set val(rp) DSR
set val(nn) 100
set val(x) 670
set val(y) 670
set val(stop) 200.0

set val(tr) /root/sankar/test/sandsr.tr
set val(nam) /root/sankar/test/sandsr.nam
set val(traffic) "/root/ns-allinone-2.1b8a/ns-2.1b8a/tcl/mobility/scene/test-cbr-100-5"
set val(movement) "/root/ns-allinone-2.1b8a/ns-2.1b8a/tcl/mobility/scene/mv-100-5"
```

The command *set* sets the values to the parameters. For example *set val(ifqlen) 50*, sets the value of the queue length to 50. In an ad hoc network, every node maintains a queue to handle packets. As the queue length is set to 50, during the

simulation the queue can handle a maximum of 50 packets only .If the queue is full, then the node starts dropping packets.

The simulation is run for a number of nodes varying from 25 to 100 with an increment of 25 nodes, for a traffic pattern of 5bps (normal traffic), 50bps and 120(bursty traffic) bps, and for a mobility rate of 1m/s, 5m/s, 10m/s and 25m/s. When the network size is varied, the traffic pattern and the mobility rate are fixed at 5 bps and 5m/s respectively. When the traffic pattern is varied, the network size is fixed at 50 and the mobility rate at 5 m/s. When the mobility rate is varied, the network size is fixed at 50 and the traffic pattern is fixed at 5bps.

The other parameters that are fixed during the entire simulation are,

Channel -----Wireless Channel
Propagation Model -----TwoRayGround
Antenna-----OmniAntenna
Queue-----DropTail/PriQueue
Queue Length-----50
Physical Layer-----WirelessPhysical
Mac-----802.11

The simulations are run in a grid size of 670X670 and for a time period of 200 seconds with the fixed bandwidth of 1 MBps. The initial energy of the battery is set at 20 J, the transmission power at 5 milliwatts and the receiving power at 1 milliwatt.

The LossModel is induced in to the network as shown below.

```
proc UniformErr { } {  
  set loss_module [new ErrorModel]  
  $loss_module set rate_ 0.3  
  $loss_module unit packet
```

```
$loss_module ranvar [new RandomVariable/Uniform]
return $loss_module
}
```

Above specified function, induces the packet losses in to the network. Changing the value for the set rate function can vary the percentage packet drop. The drop can be defined on the packet or on time. In the present work, it is defined on packet.

The mobility and the traffic pattern are created using the facility provided by NS. The random traffic connections of TCP type or CBR type can be created using the *cbrgen.tcl* file, which comes with NS2, and the node movement scenarios can be generated using *setdest* command in NS. In the present simulation CBR (constant bit rate traffic) is used. Twelve types of different traffic patterns are created for the varying network size and four types of movement patterns files are created.

When ever a simulation is run NS writes the simulation results in to two files. They are called *trace* file and *nam* file. The *nam* file can be given as an input to the Network Animator tool, which comes as a part of NS2, to view the simulation graphically. The *trace* file can be used to analyze the simulation results. Each time a simulation is run the results are written to the trace file specified.

The trace file is studied properly and routines are written to calculate the end-to-end delay, delivery rate, overhead ratio and percentage bandwidth utilization, which are discussed in §4.2

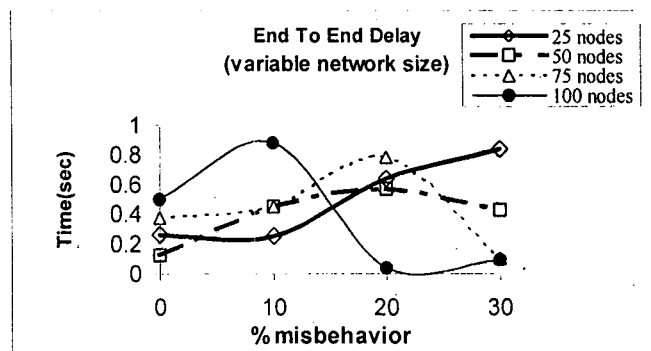
5.2 Simulation Results

Once the simulation is done and the results are drawn now it is the time to analyze the effect of misbehavior on the performance of the algorithm based on the parameters for evaluation. This section explains the effects.

5.2.1 End to End Delay

➤ For variable network size

Based on the values calculated for end-to-end delay for a node misbehavior ranging from 0% to 30% with an increment of 10%, a graph is drawn between the percentage misbehavior on the X-axis and the end-to-end delay in seconds on Y-axis, for a varying network size of 25, 50, 75 and 100 nodes. The graph drawn is shown below.

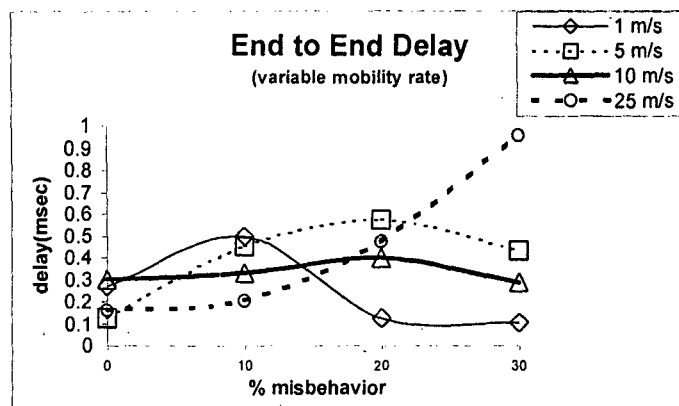


Graph5.1. End-to-End Delay Vs Percentage of misbehaving nodes for variable network size.

In the above graph, the delay increases uniformly as the percentage misbehavior increases up to a certain threshold and then starts decreasing. The threshold depends on the size of the network. For a network of size 50 and 75 the threshold is 20% misbehavior and for a network size of 100 nodes the threshold is 10%. In case of a network of 25 nodes the delay increases exponentially. This is because the nodes are moving in a fixed grid. As the network size increases the nodes are closer and closer to one another, and even if the percentage misbehavior increases the packets will take the other routes to their destination. In the case of a dense network, the delay decreases.

➤ **For variable mobility rate**

Based on the values calculated for end-to-end delay for a node misbehavior ranging from 0% to 30% with an increment of 10%, a graph is drawn between the percentage misbehavior on the X-axis and the end-to-end delay in seconds on Y-axis, for a varying mobility rate of 1m/s, 5 m/s 20 m/s and 25 m/s. The graph drawn is shown below.

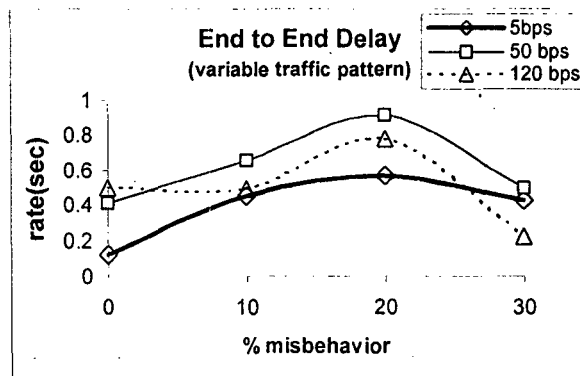


Graph5. 2 : End-to-End Delay Vs Percentage of misbehaving nodes for variable mobility rate.

In the above graph, the delay increases uniformly as the percentage misbehavior increases up to a certain threshold and then starts decreasing. The threshold depends on the movement of nodes. For mobility rates of 5m/s and 10 m/s the threshold is 20 % misbehavior. As the network movement is almost nil, i.e. at 1m/s and the network is stable, the end to end delay increases up to 10% misbehavior, then decreases and then remains uniform. As the network size is fixed and if the node movement rate is increased the nodes move rapidly and hence as the rate increases, the delay increases.

➤ **For variable traffic pattern**

Based on the values calculated for end-to-end delay for a node misbehavior ranging from 0% to 30% with an increment of 10%, a graph is drawn between the percentage misbehavior on the X-axis and the end-to-end delay in seconds on Y-axis, for a varying traffic pattern of 5pkts/sec, 50pkts/sec, and 120 pkts/sec. The graph drawn is shown below.



Graph5. 3 : End-to-End Delay Vs Percentage of misbehaving nodes for variable traffic pattern.

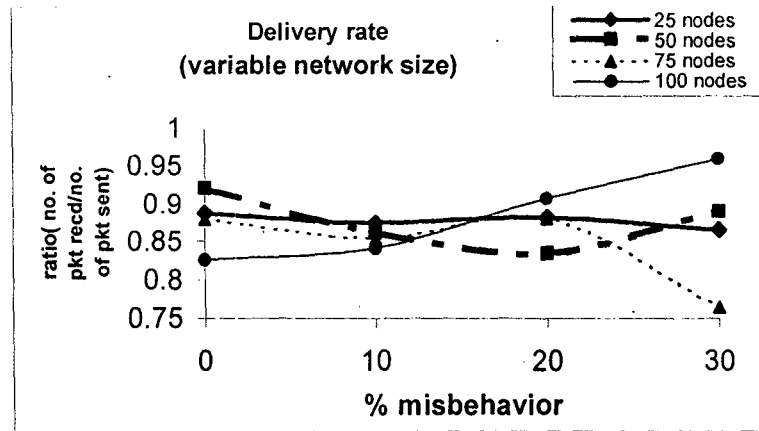
Generally if the traffic is burst, the delay increases. If the no of misbehaving nodes increases, the delay increases. In the above graph, the delay is increasing up to 20% node misbehavior. After that the delay is decreasing. So it can be said that the threshold value has been attained at 20% node misbehavior.

5.2.2 Delivery Rate

➤ **For varying Network Size :**

Based on the values calculated for delivery rate for a node misbehavior ranging from 0% to 30% with an increment of 10%, a graph is drawn between the percentage misbehavior on the X-axis and the delivery rate on

Y-axis, for a varying network size of 25 nodes, 50 nodes, 75 nodes, and 100 nodes. The graph drawn is shown below.



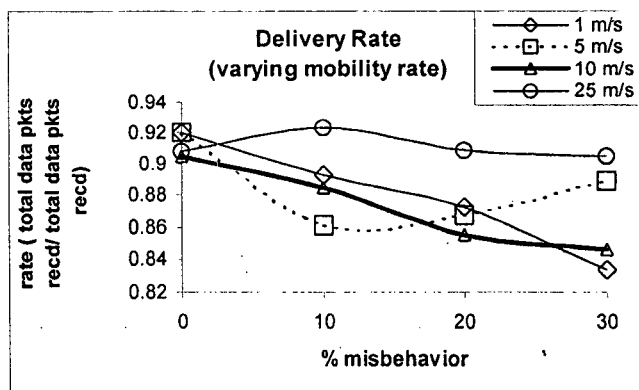
Graph 5. 4 :Delivery rate Vs Percentage of misbehaving nodes for variable network size.

In the above graph, when the network size varies from 25 nodes to 100 nodes, the delivery rate is varying. The network shows no significant trend. For a network of 25 and 100 nodes the value declines after 20% misbehavior and for a network of nodes 50 and 75 the value increases after 20% misbehavior. The value is very small for 100 nodes network, this may be due to congestion in the network. In the case of 25 nodes network the nodes may be scattered and thus, as the no of misbehaving nodes increases the rate decreases. The increment in the values of the network of 50 and 75 nodes may be due to the availability of alternative route as the network is growing. Even though some nodes are dropping packets, the other available routes may be passing them to the destination, which may lead to an increase in the value

➤ **For varying mobility rate :**

Based on the values calculated for delivery rate for a node misbehavior ranging from 0% to 30% with an increment of 10%, a graph is drawn between the percentage misbehavior on the X-axis and the delivery rate on

Y-axis, for a varying mobility rate of 1m/s, 5 m/s 20 m/s and 25 m/s. The graph drawn is shown below.



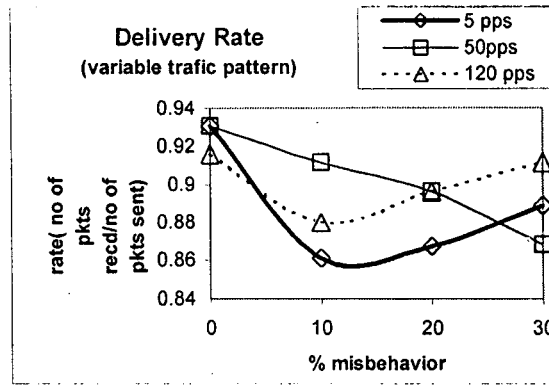
Graph5.5 : Delivery rate Vs Percentage of misbehaving nodes for variable mobility

In the above graph, a sharp decline in the delivery rate can be seen for the nodes moving with a rate of 10m/s and 1m/s. Delivery rate decreases as the number of misbehaving nodes increases. For 25 m/s mobility rate, the value increases up to 10% misbehavior and then declines. If the nodes move with a rate of 25 m/s then there is a possibility for the network partition, which will lead to the decrease in the results.

➤ **For variable Traffic Pattern**

Based on the values calculated for delivery rate for a node misbehavior ranging from 0% to 30% with an increment of 10%, a graph is drawn between the percentage misbehavior on the X-axis and the delivery rate on Y-axis, for a varying traffic pattern of 5pkts/sec, 50pkts/sec, and 120 pkts/sec. The graph drawn is shown on the next page.

In the graph shown, the delivery rate decreases when the percentage of misbehavior is 10 and then increases uniformly. For bursty traffic as the percentage of misbehavior increases the delivery rate also increasing. In case of normal traffic of 5pps(packets per second), the delivery rate decreases with increase in the % misbehavior.

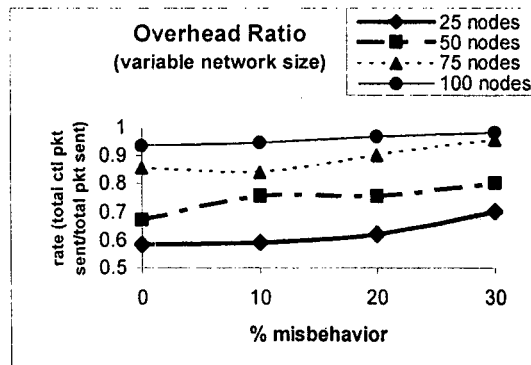


Graph5.6 : Delivery rate Vs Percentage of misbehaving nodes for variable traffic pattern

5.2.3 Overhead Ratio

➤ For varying network size

Based on the values calculated for overhead ratio for a node misbehavior ranging from 0% to 30% with an increment of 10%, a graph is drawn between the percentage misbehavior on the X-axis and the overhead ratio on Y-axis, for a varying network size of 25, 50, 75 and 100 nodes. The graph drawn is shown below.



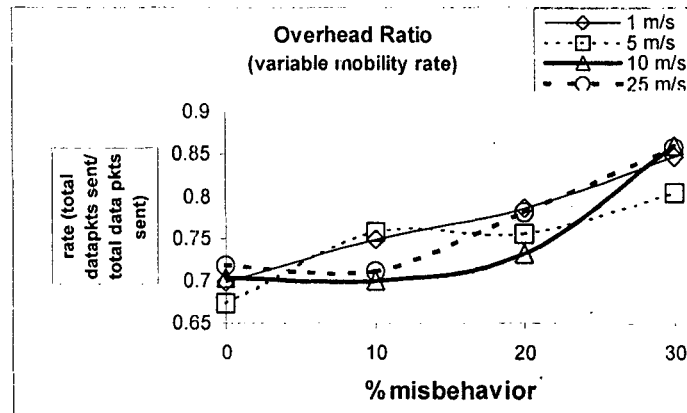
Graph5.7. Overhead Ratio Vs Percentage of misbehaving nodes for variable network size.

In the above graph it can be viewed clearly that the overhead ratio uniformly increases with the increase in the network size and increase in the percentage misbehavior of nodes. This is because of the overhead in

the DSR algorithm because the algorithm appends the source route to the data packets being transmitted.

➤ **For the varying mobility rate**

Based on the values calculated for overhead ratio for a node misbehavior ranging from 0% to 30% with an increment of 10%, a graph is drawn between the percentage misbehavior on the X-axis and the overhead rate on Y-axis, for a varying mobility rate of 1m/s, 5 m/s 20 m/s and 25 m/s. The graph drawn is shown below.

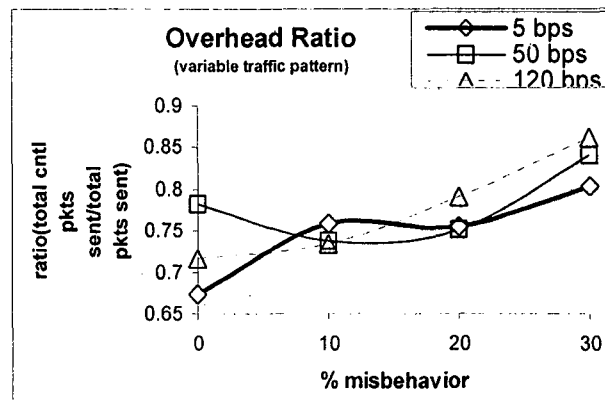


Graph5.8. Overhead Ratio Vs Percentage of misbehaving nodes for mobility rate.

In the above graph, it can easily be seen that the overhead ratio increases as the percentage misbehaving nodes increases. In the above graph, for a rate of 1m/s, the overhead ratio increases uniformly, then for 10 m/s and 20m/s it increases exponentially. As the nodes in the network move rapidly, the links may be failed and the route initiation process has to be taken place again, which results in high overheads. No significant change has been noticed for a rate of 5 m/s.

➤ **For the varying traffic pattern**

Based on the values calculated for overhead ratio for a node misbehavior ranging from 0% to 30% with an increment of 10%, a graph is drawn between the percentage misbehavior on the X-axis and the overhead rate on Y-axis, for a varying traffic pattern of 5 pps, 50 pps, 120 pps. The graph drawn is shown below.



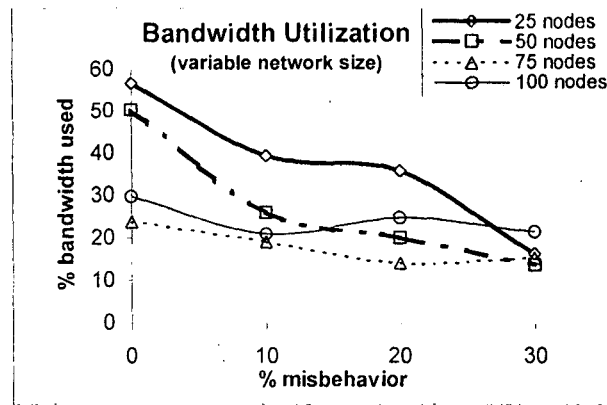
Graph5.9. Overhead Ratio Vs Percentage of misbehaving nodes for traffic pattern.

In the above graph, it can easily be seen that the overhead ratio increases as the percentage misbehaving nodes increases, with increase in the traffic pattern. As the traffic increases, the overhead also increases, due to the source routing of the algorithm.

5.2.4 Percentage bandwidth utilization

➤ **For varying network size :**

Based on the values calculated for the percentage bandwidth utilization for a node misbehavior ranging from 0% to 30% with an increment of 10%, a graph is drawn between the percentage misbehavior on the X-axis and the percentage bandwidth utilization on Y-axis, for a varying network size of 25, 50, 75 and 100 nodes. The graph drawn is shown below.

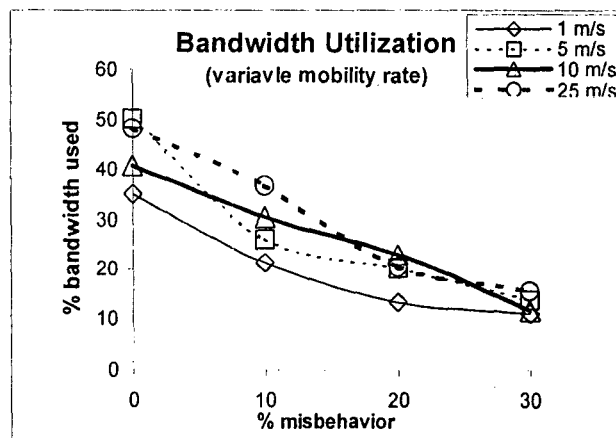


Graph5.10. Bandwidth utilized Vs Percentage of misbehaving nodes for varying network size

In the above graph, the percentage bandwidth utilization decreases as the no of nodes in the network increases as well as the percentage of misbehavior increases. The graph shows a continuous drop in the values of bandwidth utilized, for network of size 25 nodes, 50 nodes, 75 nodes. The results for the network of 75 nodes, shows varied performance.

➤ **For varying mobility pattern :**

Based on the values calculated for percentage bandwidth utilization for a node misbehavior ranging from 0% to 30% with an increment of 10%, a graph is drawn between the percentage misbehavior on the X-axis and the percentage bandwidth utilized on Y-axis, for a varying mobility rates of 1 m/s, 5 m/s, 10 m/s, 25 m/s. The graph drawn is shown below.

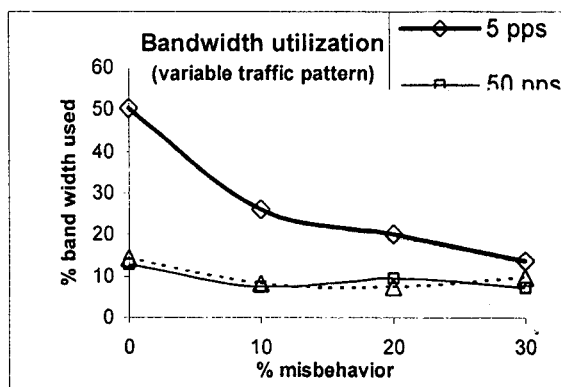


Graph5.11. Percentage bandwidth utilized Vs Percentage of misbehaving nodes for mobility rate.

In the above graph, the percentage bandwidth utilization decreases as mobility of the nodes in the network increases as well as the percentage of misbehavior increases. The graph shows a continuous drop in the values of bandwidth utilized, for the mobility rates of 1 m/s, 5 m/s, 10 m/s, 25 m/s.

➤ **For traffic pattern :**

Based on the values calculated for percentage bandwidth utilized, for a node misbehavior ranging from 0% to 30% with an increment of 10%, a graph is drawn between the percentage misbehavior on the X-axis and the percentage bandwidth utilized on Y-axis, for a varying traffic pattern of 5 pps, 50 pps, 120 pps. The graph drawn is shown below.



Graph5.12. Bandwidth utilized Vs Percentage of misbehaving nodes for traffic pattern.

In the above graph, the percentage bandwidth utilization decreases as traffic in the network and the percentage of misbehavior increases. The graph shows a continuous drop in the values of bandwidth utilized, for the traffic rates of 5 pps, 50 pps, 120 pps.

Conclusion and Future Work

In the present work, attempts have been made to simulate routing misbehavior of DSR algorithm in a mobile ad hoc network. In the context of routing misbehavior, end-to-end delay, bandwidth, overhead ratio and delivery rate, have been evaluated.

The behavior of DSR algorithm in case of routing misbehavior has shown varying results. In the cases where regular trends of increase or decrease in above-mentioned parameters are found, it has been easier to make inferences about the behavior of the algorithm. In other cases where irregular trends are found, it has been very difficult to make definite inferences. Nevertheless, attempt has been made to reason about such trends. This can be verified only by analyzing the network topology in network animator, which can give a more insight in to the changing scenarios of the topology for various simulation data.

As no project work, is completed in its entirety and even the partial completion is never with out limitations in limited time period, the present work is also not an exception. The following are the limitations of the present work:

- The source code of NS, could not been modified to incorporate routing misbehavior on a node due to the time constraints and large size of the NS code.
- The evaluation has been done only above-mentioned four parameters.

The limitations of the present work can be overcome by extending it in future. The node misbehavior can be introduced on the node to see the true misbehavior of the algorithm. The performance of the algorithm can be

analyzed for the other parameters that have not been considered in the scope of present work.

Bibliography

- [1] Andrew S. Tanenbaum, “ *Computer Networks* ”, PHI publications, 3rd edition 2000.

- [2] Charles E. Perkins, Elizabeth M. Royer, Samir R. Das, “*Ad Hoc on Demand Distance Vector (AODV) Routing*”, Internet Draft, March 2001, www.ietf.org/internet-drafts/draft-ietf-manet-aodv-08.txt.

- [3] Charles E. Perkins, P. Bhagwat, “*Highly dynamic DSDV for Mobile computer*”, SIGCOM' 94 conference on Communication Architecture, protocols and Applications
www.cs.umd.edu/projects/meml/papers/sigcomm94.ps

- [4] Charles E. Perkins, “*Mobility support, Mobile IP and Wireless Channel Support for ns-2*”, www.svrloc.org/~charliep/mobins2

- [5] Charles E. Perkins, “*Mobile Ad hoc Networking Terminology*”, Internet Draft from IETF MANET Working Group,
www.ietf.org/internet-drafts/draft-ietf-manet-term-01.txt

- [6] David B Johnson, David A. Maltz, Yih-Chun Hu , Jorjeta G. Jetcheva “*The Dynamic Source Routing algorithm for Mobile Ad hoc networks*” , Internet Draft from IETF MANET Working Group,
www.ietf.org/internet-drafts/draft-ietf-manet-dsr-05.txt

- [7] J. Maker, S. Corson, “*Mobile Ad Hoc Networks (MANET): Routing Protocol Performance Issues and Evaluation considerations*”, RFC 2501,
www.ietf.org/rfc/rfc2501.txt

- [8] Jae Chung, Mark Claypool, “*NS by Example*”, WPI Computer Science,
www.nile.wpi.edu/NS.

- [9] J. Broch, D. Maltz, D. Johnson, Y. HU, J. Jetcheva, "*A Performance Comparison of Multi-hop Ad-hoc Network Routing Protocols*", ACM/IEEE International Conference on Mobile Computing and Networking, Oct 1998
- [10] Jochen H Shiller, "*Mobile Communications*", PHI publications
- [11] Kevin Fall, Kannan Varadhan, "*Ns Manual*", VINT project, May 2001, www.mach.cs.berkeley.edu/ns/ns-man.html.
- [12] Mark Gries, "*Tutorial for Network Simulator NS-2*", www.isi.edu/nsnam/ns/tutorial
- [13] Nitin H. Vaidya. Texas A & M University "*Mobile Ad-Hoc Networks: Routing, Mac and Transport issues*", www.cs.tamu.edu/faculty/vaidya
- [14] Santhosh R. Thampuran, "*Routing Protocols for AD Hoc Networks of Mobile Nodes*", www.unix.ecs.umass.edu/~sthampur/Papers/AdhocRouting.pdf (Mobicom 98).
- [15] Sergio Marti, T. J. Giuli, Kevin Lai, Mary Baker, "*Mitigating Routing Misbehavior in Mobile Ad Hoc Networks*", MOBICOM 2000.
- [16] V Park, S Corson, "*Internet Draft for TORA*", IETF MANET working group, July, 2001, www.ietf.org/internet-drafts/draft-ietf-manet-tora-spec-04.txt

Appendix A

Packet Formats for DSR Algorithm

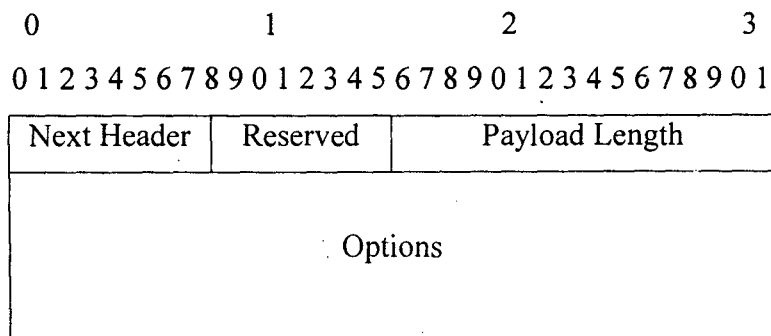
In the present topic, the various packet formats of the DSR algorithm are studied

DSR Header Format

The Dynamic Source Routing protocol makes use of a special header, carrying control information that can be included in any existing IP packet. This DSR header in a packet contains a small fixed-sized, 4-octet portion, followed by a sequence of zero or more DSR options carrying optional information. The end of the sequence of DSR options in the DSR header is implied by total length of the DSR header. For IPv4, the DSR header must immediately follow the IP header in the packet. To add a DSR header to a packet, the DSR header is inserted following the packet's IP header, before any following header such as a traditional (e.g., TCP or UDP) transport layer header. Specifically, the Protocol field in the IP header is used to indicate that a DSR header follows the IP header, and the Next Header field in the DSR header is used to indicate the type of protocol header (such as a transport layer header) following the DSR header. If any headers follow the DSR header in a packet, the total length of the DSR header (and thus the total, combined length of all DSR options present) must be a multiple of 4 octets. This requirement preserves the alignment of these following headers in the packet.

Fixed Portion of DSR Header

The fixed portion of the DSR header is used to carry information that must be present in any DSR header. This fixed portion of the DSR header has the following format:



➤ Next Header

8-bit selector. Identifies the type of header immediately following the DSR header. Uses the same values as the IPv4 Protocol field

➤ Reserved

Must be sent as 0 and ignored on reception.

➤ Payload Length

The length of the DSR header, excluding the 4-octet fixed portion. The value of the Payload Length field defines the total length of all options carried in the DSR header.

➤ Options

Variable-length field; the length of the Options field is specified by the Payload Length field in this DSR header. Contains one or more pieces of optional information (DSR options), encoded in type-length-value (TLV) format (with the exception of the Pad1 option).

The placement of DSR options following the fixed portion of the DSR header may be padded for alignment. However, due to the typically limited available wireless bandwidth in ad hoc networks, this padding is not required, and receiving nodes must not expect options within a DSR header to be aligned.

A node inserting a DSR header into a packet MUST set the Don't Fragment (DF) bit in the packet's IP header.

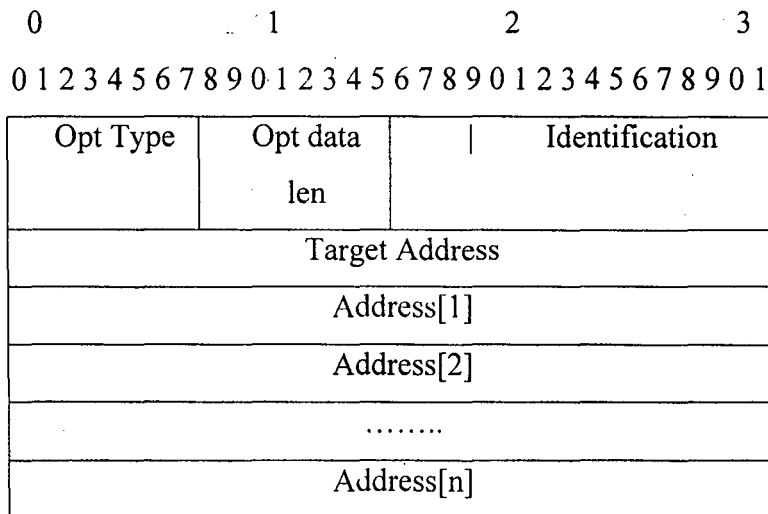
The following types of DSR options that are defined in the Internet draft for use within a DSR header:

- Route Request option
- Route Reply option
- Route Error option
- Acknowledgement Request option
- Acknowledgement option
- DSR Source Route option
- Pad1 option
- PadN option

In the present appendix only the first three formats are discussed.

Route Request Option:

The Route Request option in a DSR header is encoded as follows:



IP fields:

- Source Address
MUST be set to the address of the node originating this packet. Intermediate nodes that retransmit the packet to propagate the Route Request MUST NOT change this field.
- Destination Address
Must be set to the IP limited broadcast address (255.255.255.255).
- Hop Limit (TTL)
May be varied from 1 to 255, for example to implement non-propagating Route Requests and Route Request expanding-ring searches (Section 3.3.4).

Route Request fields:

Option Type 2

Opt Data Len

8-bit unsigned integer. Length of the option, in octets, excluding the Option Type and Opt Data Len fields.

Identification

A unique value generated by the initiator (original sender) of the Route Request. This value allows a receiving node to determine whether it has recently seen a copy of this Route Request: if this Identification value is found by this receiving node in its Route Request Table, this receiving node must discard the Route Request. When propagating a Route Request, this field must be copied from the received copy of the Route Request being propagated.

Target Address

The address of the node that is the target of the Route Request.

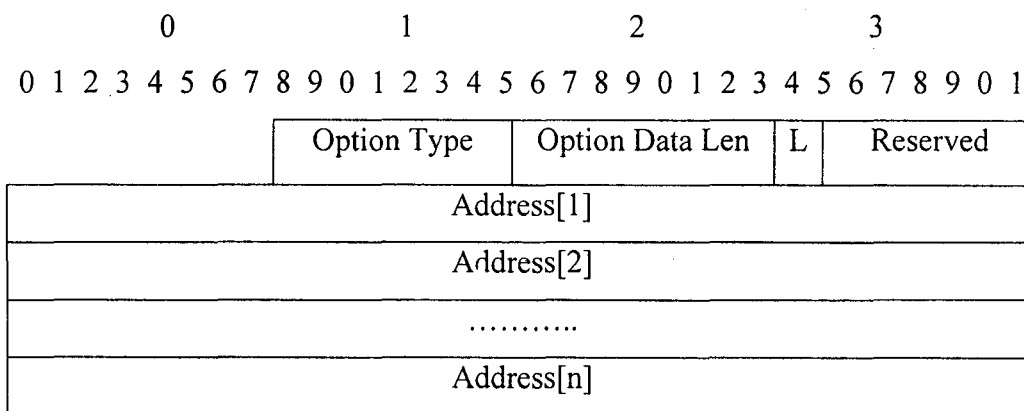
Address[1..n]

Address[i] is the address of the i^{th} node recorded in the Route Request option. The address given in the Source Address field in the IP header is the address of the initiator of the Route Discovery and must not be listed in the Address[i] fields; the address given in Address[1] is thus the address of the first node on the path after the initiator. The number of addresses present in this field is indicated by the Opt Data Len field in the option ($n = (\text{Opt Data Len} - 6) / 4$). Each node propagating the Route Request adds its own address to this list, increasing the Opt Data Len value by 4 octets.

The Route Request option must not appear more than once within a DSR header.

Route Reply Option

The Route Reply option in a DSR header is encoded as follows:



IP fields:

- Source Address

Set to the address of the node sending the Route Reply. In the case of a node sending a reply from its Route Cache or sending a gratuitous Route Reply, this address can differ from the address that was the target of the Route Discovery.

➤ Destination Address

MUST be set to the address of the source node of the route being returned. Copied from the Source Address field of the Route Request generating the Route Reply, or in the case of a gratuitous Route Reply, copied from the Source Address field of the data packet triggering the gratuitous Reply.

➤ Route Reply fields:

Option Type 3

Opt Data Len 8-bit unsigned integer. Length of the option, in octets, excluding the Option Type and Opt Data Len fields.

➤ Last Hop External (L)

Set to indicate that the last hop given by the Route Reply (the link from Address[n-1] to Address[n]) is actually an arbitrary path in a network external to the DSR network; the exact route outside the DSR network is not represented in the Route Reply. Nodes caching this hop in their Route Cache must flag the cached hop with the External flag. Such hops must not be returned in a cached Route Reply generated from this Route Cache entry, and selection of routes from the Route Cache to route a packet being sent MUST prefer routes that contain no hops flagged as External.

➤ Reserved

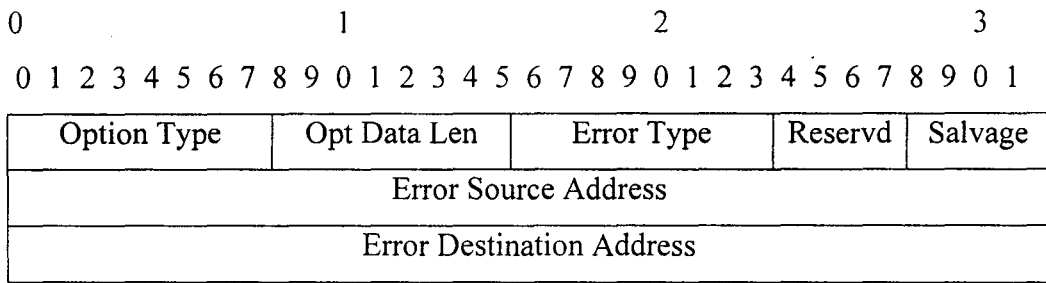
Must be sent as 0 and ignored on reception.

➤ Address[1..n]

This is same as described in the Route Request option above.

Route Error Option

The Route Error option in a DSR header is encoded as follows:



Type-Specific Information

Option Type 4

Opt Data Len 8-bit unsigned integer. Length of the option, in octets, excluding the option Type and Opt Data Len fields.

For the current definition of the Route Error option, this field must be set to 10, plus the size of any Type-Specific Information present in the Route Error. Further extensions to the Route Error option format may also be included after the Type-Specific Information portion of the Route Error option specified above. The Opt Data Len field will indicate the presence of such extensions. When the Opt Data Len is greater than that required for the fixed portion of the Route Error plus the necessary Type-Specific Information as indicated by the Option Type value in the option, the remaining octets are interpreted as extensions. Currently, no such further extensions have been defined.

➤ Error Type

The type of error encountered. Currently, the following type value is defined:

1 = NODE_UNREACHABLE

Other values of the Error Type field are reserved for future use.

➤ Reservd

Reserved. Must be sent as 0 and ignored on reception.

➤ Salvage

A 4-bit unsigned integer. Copied from the Salvage field in the DSR Source Route option of the packet triggering the Route Error.

The "total salvage count" of the Route Error option is derived from the value in the Salvage field of this Route Error option and all preceding Route Error options in the packet as follows: the total salvage count is the sum of, for each such Route Error option, one plus the value in the Salvage field of that Route Error option.

➤ Error Source Address

The address of the node originating the Route Error (e.g., the node that attempted to forward a packet and discovered the link failure).

➤ Error Destination Address

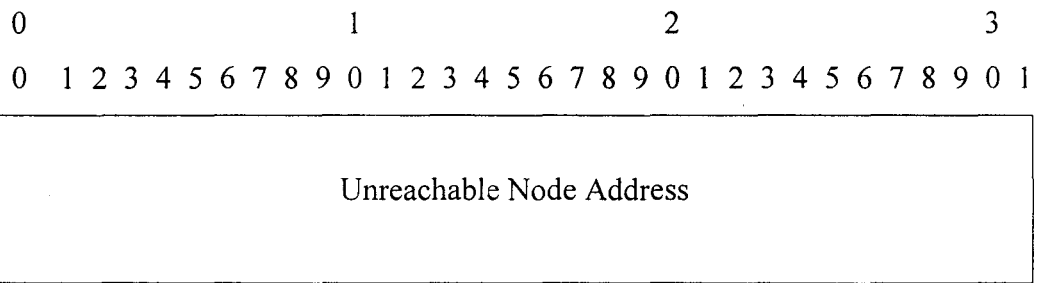
The address of the node to which the Route Error must be delivered. For example, when the Error Type field is set to `NODE_UNREACHABLE`, this field will be set to the address of the node that generated the routing information claiming that the hop from the Error Source Address to Unreachable Node Address (specified in the Type-Specific Information) was a valid hop.

➤ Type-Specific Information

Information specific to the Error Type of this Route Error message. Currently, the Type-Specific Information field is defined only for Route Error messages of type `NODE_UNREACHABLE`.

In this case, the

Type-Specific Information field is defined as follows:



➤ Unreachable Node Address

The address of the node that was found to be unreachable (the next-hop neighbor to which the node with address Error Source Address was attempting to transmit the packet).

A Route Error option may appear one or more times within a DSR header.

Tabular data of Simulation Results

The following are the results obtained at the end of simulations.

1. The following are the simulation results for varying network size keeping the mobility rate and traffic pattern constant, respectively at 5m/s and 5 packets per second.

For a network of 25 nodes

% Routing misbehavior	0%	10%	20%	30%
End to end delay	0.264302	0.255228	0.640460	0.839489
No of pkt recd(r)	1561	1276	1062	715
No of pkt sent (s)	1758	1459	1205	826
Delivery rate (r/s)	0.88794	0.87457	0.88132	0.865617
DSR control pkt (c)	4929	4165	4733	6401
Ack pkt (a)	8737	7064	5082	3776
Total data pkt (d)	9644	7740	5968	4324
overhead ratio (c+a)/d	0.586271	0.591944	0.6218716	0.701813
Bandwidth used	56.64542	39.38808	35.815695	16.085

For a network of 50 nodes

% routing misbehavior	0%	10%	20%	30%
End to end delay	0.123494	0.452844	0.570349	0.431075
No of pkt recd(r)	1692	1307	1055	778
No of pkt sent (s)	1838	1518	1216	875
Delivery rate (r/s)	0.9205658	0.8610013	0.8347039	0.8891242
DSR control pkt (c)	12011	18337	14091	14351
Ack pkt (a)	9326	6956	5506	4056
Total data pkt (d)	10330	8056	6341	4502
overhead ratio (c+a)/d	0.6737929	0.7584335	0.7555324	0.8034833
Bandwidth used	50.22496	25.939175	20.039368	13.671514

For a network of 75 nodes

% routing misbehavior	0%	10%	20%	30%
End to end delay	0.376188	0.449949	0.782632	.104134
No of pkt recd(r)	1240	1069	805	392
No of pkt sent (s)	1410	1253	951	513
Delivery rate (r/s)	0.8794326	0.8531524	0.8797814	0.76413255
DSR control pkt (c)	44088	32239	46819	59223
Ack pkt (a)	7437	6401	4527	2047
Total data pkt (d)	8645	7279	5323	2650
overhead ratio (c+a)/d	0.8563237	0.8414817	0.90606857	0.95854192
Bandwidth used	23.823548	19.022790	13.992096	15.297479

For a network of 100 nodes

% routing misbehavior	0%	10%	20%	30%
End to end delay	0.502403	0.882866	0.043697	0.099480
No of pkt recd(r)	1670	1364	738	290
No of pkt sent (s)	2032	1664	1035	553
Delivery rate (r/s)	0.82185039	0.81971154	0.7130434	0.52441229
DSR control pkt (c)	127366	127440	126737	138875
Ack pkt (a)	7609	5986	3157	1086
Total data pkt (d)	9097	7087	4002	1848
overhead ratio (c+a)/d	0.93685795	0.94956553	0.970111131	0.986868387
Bandwidth used	29.794615	20.920955	24.709252	21.486650

2. The following are the simulation results for varying traffic pattern keeping the mobility rate and network size constant, respectively at 5m/s and 50 nodes second.

For a traffic pattern of 5 packets/second

% routing misbehavior	0%	10%	20%	30%
End to end delay	0.123494	0.452844	0.570349	0.431075
No of pkt recd(r)	1692	1307	1055	778
No of pkt sent (s)	1838	1518	1216	875
Delivery rate (r/s)	0.93056583	0.86100131	0.86759868	0.88912856
DSR control pkt (c)	12011	18337	14091	14351
Ack pkt (a)	9326	6956	5506	4056
Total data pkt (d)	10330	8056	6341	4502
overhead ratio (c+a)/d	0.6737929	0.7584335	0.75553242	0.8034833
Bandwidth used	50.224960	25.939275	20.039368	13.671514

For a traffic pattern of 50 packets/second

%of Misbehaving nodes	0%	10%	20%	30%
End to end delay	0.415037	0.654688	0.913433	0.502701
No of pkt recd(r)	2124	1509	1135	705
No of pkt sent (s)	2283	1655	1266	812
Delivery rate (r/s)	0.93035479	0.91178248	0.89652449	0.86822660
DSR control pkt (c)	15501	15729	13239	18463
Ack pkt (a)	10906	7581	5976	3670
Total data pkt (d)	11220	8355	6360	4196
overhead ratio (c+a)/d	0.78180987	0.7361440	0.75131965	0.84063200
Bandwidth used	12.874100	7.458136	9.301938	7.244039

For a traffic pattern of 120 packets/second

% routing misbehavior	0%	10%	20%	30%
End to end delay	0.5011221	0.493438	0.780764	0.229068
No of pkt recd(r)	2216	1668	1070	723
No of pkt sent (s)	2420	1896	1194	793
Delivery rate (r/s)	0.91570248	0.87974684	0.89614740	0.91172762
DSR control pkt (c)	17688	16639	16964	21759
Ack pkt (a)	10856	8412	5573	3832
Total data pkt (d)	11277	1993	5939	4119
overhead ratio (c+a)/d	0.71680771	0.7358418	0.79143840	0.8613598
Bandwidth used	14.442447	8.126614	7.352433	19.662387

3. The following are the simulation results for varying mobility rate keeping the traffic pattern and network size constant, respectively at 5 packets per second and 50 nodes second.

For a mobility rate of 1 m/s

% routing misbehavior	0%	10%	20%	30%
End to end delay	0.268889	0.496488	.123466	0.104748
No of pkt recd(r)	1555	1286	913	609
No of pkt sent (s)	1729	1440	1046	730
Delivery rate (r/s)	0.92056583	0.8930556	0.87284895	0.8342466
DSR control pkt (c)	14947	16249	0.16747	19021
Ack pkt (a)	9450	7875	5748	3821
Total data pkt (d)	10511	8095	6130	4142
overhead ratio (c+a)/d	0.69889423	0.74875074	0.78585153	0.84650163
Bandwidth used	35.034094	21.243725	13.269155	11.070335

For a mobility rate of 5 m/s

% routing misbehavior	0%	10%	20%	30%
End to end delay	0.123494	0.452844	0.570349	0.104748
No of pkt recd(r)	1692	1307	1055	609
No of pkt sent (s)	1838	1518	1216	730
Delivery rate (r/s)	0.9205658	0.86100132	0.86759868	0.8342466
DSR control pkt (c)	12011	18337	14091	19021
Ack pkt (a)	9326	6956	5506	3821
Total data pkt (d)	10330	8056	6341	4142
overhead ratio (c+a)/d	0.67379291	0.75843353	0.75553242	0.84650163
Bandwidth used	50.224960	25.939175	20.039368	11.070335

For a mobility rate of 10 m/s

% routing misbehavior	0%	10%	20%	30%
End to end delay	0.301322	0.33052	0.400321	0.290032
No of pkt recd(r)	1500	1239	973	667
No of pkt sent (s)	1657	1400	1137	788
Delivery rate (r/s)	0.90525045	0.885000	0.85576077	0.8464467
DSR control pkt (c)	14897	11971	10975	21125
Ack pkt (a)	8712	7004	5378	3386
Total data pkt (d)	9949	8118	6001	4022
overhead ratio (c+a)/d	0.70352822	0.7003654	0.73154692	0.85904041
Bandwidth used	40.716495	30.543853	23.017404	11.727117

For a mobility rate of 25 m/s

% routing misbehavior	0%	10%	20%	30%
End to end delay	0.160511	0.203078	0.472998	0.959979
No of pkt recd(r)	1638	1357	1040	733
No of pkt sent (s)	1803	1469	1144	810
Delivery rate (r/s)	0.90848586	0.92375766	0.90909091	0.90493827
DSR control pkt (c)	15223	12213	15195	19921
Ack pkt (a)	8607	7090	5088	3389
Total data pkt (d)	9324	7819	5679	3878
overhead ratio (c+a)/d	0.71876697	0.71171005	0.7812572	0.85736354
Bandwidth used	48.278344	36.619194	20.386068	15.655287